Benefits of NGINX (Advanced/Optional)

Benefits of NGINX vs PHP Built-in Server

1. Performance

Metric	PHP -S	NGINX + PHP-FPM
Concurrent Connections	1	1000+
Requests/Second	~100	10,000+
Memory Usage	High per request	Optimized
Static File Serving	Slow	Lightning fast

2. Production Features

- ✅ SSL/HTTPS Support
- ✅ Compression (gzip)
- ✅ Load Balancing
- ✅ Reverse Proxy
- ✅ Rate Limiting
- ✅ Security Headers

Benefits: Real-World Scenarios

Scenario 1: Multiple Users

PHP -S: 10 users = slow/timeout

NGINX: 1000+ users = smooth experience

Scenario 2: Image/CSS Heavy Site

PHP -S: PHP processes every file

NGINX: Serves static files directly (10x faster)

Scenario 3: API + Frontend

PHP -S: Single point of failure

NGINX: Can route to multiple backends

Scenario 4: Security

PHP -S: Basic security only

NGINX: Production-grade security features

Benefits: Static File Handling

PHP Built-in Server

```
Browser → PHP -S → Check if PHP file → Serve file
```

Every file goes through the PHP parser

NGINX

```
Browser → NGINX → Is it .php? → Yes: PHP-FPM
                              → No: Serve directly
```

Static files bypass PHP entirely

Performance Impact

- **Images**: 10x faster with NGINX
- **CSS/JS**: 5x faster with NGINX
- **Downloads**: No timeout with NGINX

Benefits: Real Production Features

1. HTTPS/SSL Support

```
server {
    listen 443 ssl;
    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/key.pem;

    # Your PHP configuration here
}
```

2. Compression

```
gzip on;
gzip_types text/plain text/css application/json application/javascript;
```

3. Rate Limiting

```
limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;

location /api/ {
    limit_req zone=api burst=20 nodelay;
    # PHP processing here
}
```

Benefits: Scalability

Horizontal Scaling with Load Balancing

```
upstream php_backend {
    server 127.0.0.1:9000;
    server 127.0.0.1:9001;
    server 127.0.0.1:9002;
}

server {
    location ~ \.php$ {
        fastcgi_pass php_backend;
        # Other fastcgi settings
    }
}
```

Microservices Architecture

```
location /api/users/ {
    proxy_pass http://users-service:3000;
}

location /api/orders/ {
    proxy_pass http://orders-service:3000;
}

location ~ \.php$ {
    fastcgi_pass 127.0.0.1:9000;
}
```

Troubleshooting Common Issues

1. 502 Bad Gateway Error

Cause: PHP-FPM not running

Solution:

```
# Check PHP-FPM status
ps aux | grep php-fpm

# Start PHP-FPM
sudo systemctl start php-fpm  # Linux
brew services start php       # macOS
```

2. File Not Found (404)

Cause: Wrong root path or missing try_files

Solution: Check `root` directive and add:

```
location / {
    try_files $uri $uri/ =404;
}
```

3. Download PHP Files Instead of Executing

Cause: PHP location block not working

Solution: Check this block exists:

```
location ~ \.php$ {
    fastcgi_pass 127.0.0.1:9000;
    # ... other settings
}
```

Testing Your Setup

1. Create Comprehensive Test Files

performance_test.php:

```php
<?php
$start = microtime(true);
// Simulate some work
for ($i = 0; $i < 100000; $i++) {
    $data[] = md5($i);
}
$end = microtime(true);
$time = ($end - $start) * 1000; // Convert to milliseconds
header('Content-Type: application/json');
echo json_encode([
    'server' => 'NGINX + PHP-FPM',
    'processing_time_ms' => round($time, 2),
    'memory_usage_mb' => round(memory_get_usage() / 1024 / 1024, 2),
    'timestamp' => date('Y-m-d H:i:s')
]);
```

2. Load Test with cURL

```
# Test multiple requests
for i in {1..10}; do
    curl http://localhost/performance_test.php
done
```

Monitoring and Maintenance

1. Check NGINX Status

```
# Linux
sudo systemctl status nginx

# All platforms
nginx -t  # Test configuration
```

2. Check PHP-FPM Status

```
# Linux
sudo systemctl status php-fpm

# macOS
brew services list | grep php
```

3. View Error Logs

```
# NGINX errors
tail -f /var/log/nginx/error.log  # Linux
tail -f /usr/local/var/log/nginx/error.log  # macOS

# PHP-FPM errors
tail -f /var/log/php-fpm/www-error.log
```

Key Configuration Summary

Minimal Working NGINX Config

```
server {
    listen 80;
    root /var/www/html;
    index index.php index.html;

    location ~ \.php$ {
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

Commands to Remember

```
nginx -t            # Test configuration
nginx -s reload     # Reload configuration
nginx -s stop       # Stop NGINX
systemctl start php-fpm  # Start PHP-FPM (Linux)
```

Key Takeaways

Why NGINX + PHP-FPM > PHP -S

1. **Performance**: 10x better for static files, handles 1000+ concurrent users
2. **Production Features**: SSL, compression, security headers, load balancing
3. **Scalability**: Can handle real-world traffic and growth
4. **Industry Standard**: What professionals use in production
5. **Separation of Concerns**: NGINX for web serving, PHP for processing

Skills You've Gained

- Professional web server configuration
- Production-ready PHP deployment
- Performance optimization techniques
- Real-world web architecture understanding

Next: Building complete web applications with database integration!

Resources for Further Learning

Documentation

- **NGINX Beginner's Guide**: http://nginx.org/en/docs/beginners_guide.html
- **PHP-FPM Configuration**: https://www.php.net/manual/en/install.fpm.php
- **NGINX + PHP Best Practices**: https://www.nginx.com/resources/wiki/start/topics/examples/phpfcgi/

Performance Testing

- **Apache Bench**: `ab -n 1000 -c 10 http://localhost/`
- **wrk**: Modern HTTP benchmarking tool
- **Browser DevTools**: Network tab for real-world testing

You're now ready for production-level web development!